2. Interpolation

```
> restart;
```

```
> interpolin := proc(fX, x)
  local N, k, h;
  N := nops(fX) - 1 :
  if (x = 1) then
  return fX[N + 1];
  fi:
  k := floor(x·N);
  h := 1/N;
  fX[k + 1] + (fX[k + 2] - fX[k + 1])/h · (x - k/N) :
  end proc:
```

Exemple

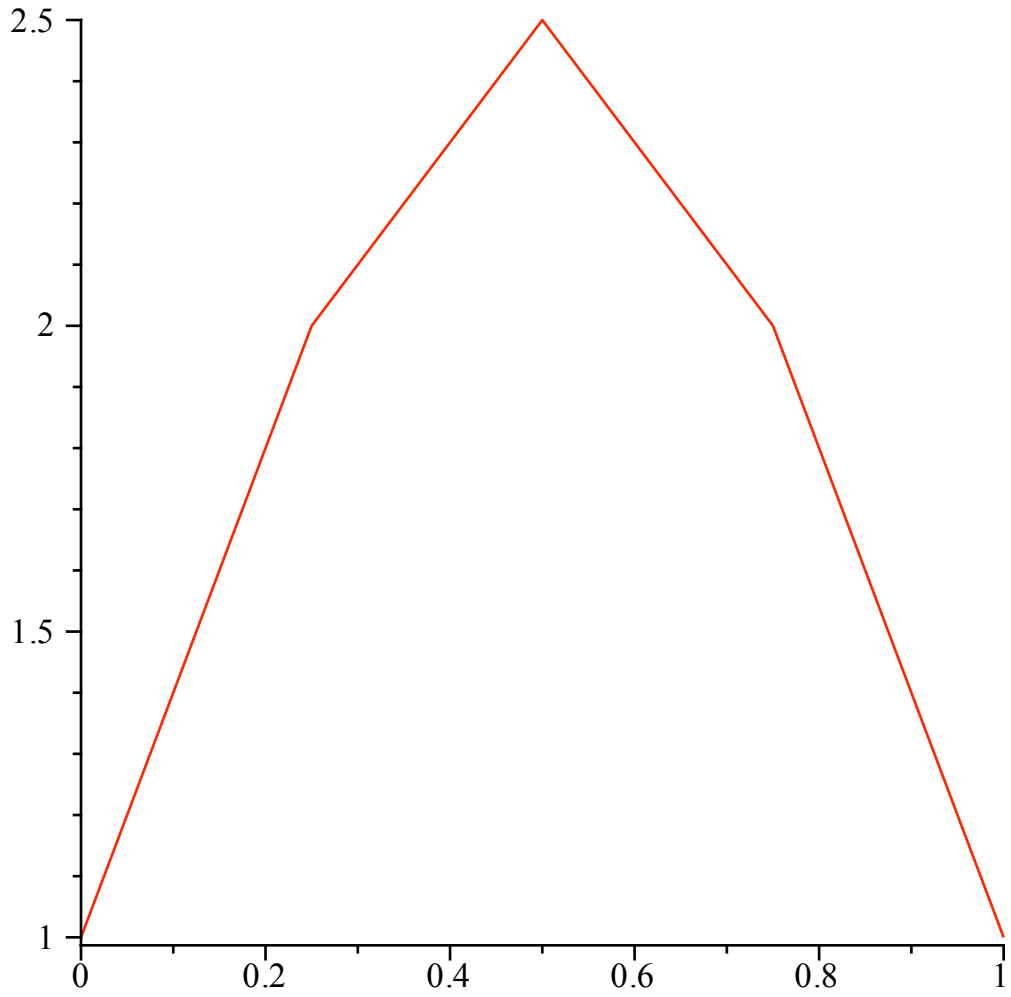
```
> interpolin([1, 2, 2.5, 2, 1], 0.3);
```

$$2.100000000 \qquad \textbf{(1.1.1)}$$

```
> interpolin([1, 2, 2.5, 2, 1], 1);
```

$$1 \qquad \textbf{(1.1.2)}$$

```
> plot([[0, 1], [0.25, 2], [0.5, 2.5], [0.75, 2], [1, 1]]);
```



2.2.3 Exercices

Exercice 4

```
> T := 23;
```

$$T := 23 \qquad \textbf{(1.2.1.1)}$$

```
> evalf(interpolin([1.308, 1.141, 1.005, 0.896, 0.804, 0.727, 0.661], (T - 10)/30));
```

$$0.9396000000 \qquad \textbf{(1.2.1.2)}$$

```
>
```

Exercice 5

```
> #Premiere methode
```

```
> interpolin2 := proc( fX, x, a, b )
  interpolin( fX, (x - a)/(b - a) ) :
  end proc:
> interpolin2( [1.308, 1.141, 1.005, 0.896, 0.804, 0.727, 0.661], T, 10, 40);
                                  0.9396000000                          (1.2.2.1)
> #Deuxieme methode
> interpolin2bis := proc( fX, x, a, b )
  local N, k, h;
  N := nops( fX ) - 1 :
  if (x = b) then
  return fX[N + 1];
  fi:
  k := floor( N·(x - a)/(b - a) );
  h := (b - a)/N;
  fX[k + 1] + (fX[k + 2] - fX[k + 1])/h ·(x - (a + k·h)) :

  end proc:
> evalf (interpolin2bis( [1.308, 1.141, 1.005, 0.896, 0.804, 0.727, 0.661], T, 10, 40) );
                                  0.9396000000                          (1.2.2.2)
>
```

Exercice 8

```
> 362.78 + (512.35 - 362.78)/(20 - 15) ·1;
                                  392.6940000                           (1.2.3.1)
> interpolin2( [0, 227.04, 362.78, 512.35, 602.97, 901.67], 16, 0, 30);
                                  462.4933333                           (1.2.3.2)
> interpolin2bis( [0, 227.04, 362.78, 512.35, 602.97, 901.67], 16, 0, 30);
                                  462.4933333                           (1.2.3.3)
> interpolin2( [362.78, 512.35], 16, 15, 20);
                                  392.6940000                           (1.2.3.4)
```

Exercice 9

```
> localisation := proc(X, x)
  local i;
  if (x = X[nops(X)]) then
  return nops(X) - 1;
```

```
    fi:
    i := 1 :
    while (X[i + 1] ≤ x) do
    i := i + 1 :
    od:
    i := i − 1;
    end proc:
> localisation([0, 0.5, 1, 1.5], 1.3);
```

2 **(1.2.4.1)**

```
>
```

Exercice 10

```
> interpolin3 := proc(X, fX, x)
    local i;
    if (x = X[nops(X)]) then
    return fX[nops(X)];
    fi:
    i := localisation(X, x);
    interpolin2([fX[i + 1], fX[i + 2]], x, X[i + 1], X[i + 2]);
    end proc:
> interpolin3([0, 10, 15, 20, 22.5, 30], [0, 227.04, 362.78, 512.35, 602.97, 901.67],
      16);
```

392.6940000 **(1.2.5.1)**

```
>
```

2.3.3 Algorithme

```
> interpolcub := proc(fX, x)
    local N, k, a, s;
    N := nops(fX) − 3 :
    if (x = 1) then
    return fX[N + 2];
    fi:
    k := floor(N·x) :
    a := x·N − k;
    s := -  fX[k + 1]·a·(1 − a)·(2 − a) / 6 :
    s := s + fX[k + 2]·(a + 1)·(1 − a)·(2 − a) / 2 :
    s := s + fX[k + 3]·(a + 1)·a·(2 − a) / 2 :
    s := s − fX[k + 4]·(a + 1)·a·(1 − a) / 6 :
```

```
end proc:
>
```

2.3.6 Exercices

Exercice 13

```
> interpolcub2 := proc( fX, x, a, b )
  interpolcul( fX, (x - a)/(b - a) ) :
  end proc:
>
```